Real-Time Communications on the Web; Considerations

For the RTC Workshop, Mountain View, October 2010

David Singer

Multimedia and Software Standards, Apple Inc.

1 Overview

This paper attempts to present some considerations for how we might go about defining a framework for handling real-time communications (telephony, possibly including video-telephony) on the web.

2 Layers

2.1 Introduction

There are 3 main parts of a (video-)telephony session.

The first is discovery: converting some identifier for a person into something that can be addressed on the internet – which means, eventually, an IP address (v4 or v6, one assumes).

The second, unfortunately, is connectivity. I say unfortunately, because the design of IP originally assumed that connectivity was simply a matter of being connected to the same network. However, the use of firewalls, and the prevalence of Network Address Translation (NAT), mean that connectivity is not assured. Indeed, it is sometimes the case that the only assured protocol is HTTP (or access to mail servers, which don't appear to be very useful for this problem area). There *are* telephony products that fall back to HTTP, but this is not ideal: the presence of proxies and caches can induce significant delay, and they are often 'transparent' (i.e. not configured at the end-points but part of the network and hence unavoidable). If RTP is used for the session, then more considerations come into play; the RTP specification requires the use of multiple UDP ports, though fairly simple non-standard techniques can ameliorate this. Typically techniques used to establish UDP/RTP connectivity include ICE and STUN, and, in the apparently odd case that the end points can talk to the public internet but not each other, TURN. The problems with NAT may be alleviated by the use of IPv6, where it is not needed; but many of these are firewall issues, which will remain.

The third part is the communication – the passing of data, both the call data (audio and possibly video), and also the meta-data – call setup information, and information that is used to manage the call in process (e.g. that can be used to estimate network jitter). Two large considerations for interoperability here are the audio and video compression formats, and the use of encryption. Cellular phone data is routinely encrypted, and given

the prevalence of (easily tapped) wireless networks, it seems fairly clear that encryption needs to be part of any interoperable framework for IP-based calling, as well.

2.2 Structure

It would be nice if the user agent (UA) could offer fairly small building blocks with which to build a web-based telephone in a variety of ways, using scripting and the like to glue the pieces together. However, the rather tight requirements on latency once data is flowing mean that such modularization is probably limited.

In the rest of this paper I imagine a structure in which there is some HTML element that represents a telephony session, perhaps rather akin to the video element in HTML5.

2.3 Discovery

People need to be able to take some 'permanent' address with them as they move around – from home to the coffee shop, to the wireless network, to work, in hotels and so on when travelling, and so forth. Address resolution is typically done in the Internet by converting a domain name into an IP address using an address resolution protocol. It is possible that some discovery mechanisms will give each person their own domain name, and implement a 'dynamic' service for supplying current IP addresses for mobile customers, but such an approach has some problems, and even if they are alleviated, it is by no means the only approach. One of the problems is that DNS is not well suited to delivering status information ('the person you are calling is currently unavailable') or handling such user-settings as 'send directly to voicemail'.

Other discovery mechanisms include converting a personal identifier to an IP address, such as an AOL screen name, or email address; or 'hitching a ride' on some other mechanism such as a telephone number.

One important use-case in which discovery is not needed (at least, not in one direction) is the provision of on-line interactive services such as customer support. In this case, the site offering the web page is also offering the destination address to call; the user does not need to 'discover' the IP location of the service centre.

For these reasons, it is perhaps best if discovery is separated. One can imagine a set of protocols that resolve some kind of persistent personal identifier into an IP-addressable entity, and handle pre-call issues such as unavailability, 'send to voicemail', and so on. However, such protocols do raise privacy issues; we perhaps do not want arbitrary web sites being able to determine 'is he on-line?', 'what is his IP address?' (and hence, 'where is he')', and so on.

As noted above, part of the protocol must be the management of data-plane privacy as well, which generally means encrypting data in transit. It's possible that general privacy measures, such as TLS, can be used here, but it is more likely that the encryption will need tailoring to the communication protocol, for efficiency. Discovery may need to include the acquisition of a public key or keys for the person addressed.

2.4 Connectivity

Connectivity is fairly intimately tied to communication – one is, after all, establishing connectivity for the communication protocol one intends to use. Hence it may well be

that the trying of direct connections, the use of ICE/STUN, falling back to proxies, TURN or TCP or HTTP tunnelling, can be left to the protocol in question.

2.5 Communication

One of the key problems in the video element has been the (so far unsuccessful) attempt to establish a set of default, or 'fall-back' codecs. Content providers are, in general, resistant to providing their content in every conceivable codec or codec combination. However, in real-time communications it is less problematic, as encoding is on-the-fly, and the early part of the communications can be an exchange of capability, including which codecs are supported at the two ends.

It is, however, a problem. We don't want phone calls to fail because of a complete codec mismatch between the two ends. Like the streaming case, the fallback codec does not have to be best in breed – and anyway, best in breed is a fleeting appellation, while any mandate will have to last forever.

The codec question is, also, moot if we don't have a standard protocol. RTP is old, and has many design assumptions that are questionable, but it is in widespread use and understood. SIP may have once been thought of as simple, and is no longer such; but again, it is in use and does the job. And these two have some momentum in both deployments and standards (from 3GPP, ITU, as well as the core specifications from the IETF).

As noted above, there is also a need for encryption on the data plane. It really should not be normal that people's conversations are easily 'sniffed'. A suitable use of public/private keys for initial setup (and the public key can come from the discovery phase, if needed) followed by cheaper symmetric schemes, within a standard framework (standard for the protocol) should suffice here.

2.6 Embedding Considerations

As noted in the invitation, we will need to deal with privacy issues. Somehow we will want to enable 'custom controllers' (HTML/CSS/JavaScript) without making it even vaguely possible that web pages can be built that 'listen in' to us without our consent or knowledge.

3 Conclusions

Overall, this is more of a survey than recommendations, but some themes emerge:

- worry about privacy generally, not just in the exposure of the camera and microphone;
- modularize as much as possible (though this paper doesn't do very much)
- use, or at least suggest, reasonable baselines for protocols and codecs, if possible.